[0049] In **FIG. 3**, an application program such as application program with identifier ID1 requests access to various resources. An embodiment of the invention receives the request and processes the request according to the privilege or access specified for the application program for the resource. In the example of **FIG. 3**, the application program has access to some operating system resources (e.g., files and settings) at a read-only privilege. If the application program sends a request for read access to one of these resources, an embodiment of the invention grants the application program read-only access to the one resource. If the application program sends a request to modify one of these read-only resources at **302**, an embodiment of the invention denies the application program access to the one resource. The request fails silently (e.g., no response returned to the application program) or explicitly (e.g., a negative response is returned to the application program) at **304**. The application program also has access to application private resources (e.g., those files and settings associated with the application program) at a read-write privilege. Because these resources are associated with the application program, modification of these resources generally does not raise operating system fragility issues. The operating system has little knowledge and interest in the semantics of these resources.

[0050] The application program has access to other operating system resources at a protected privilege. If the application program sends a request to modify one of these protected operating system resources (e.g., settings or files) at **306**, an embodiment of the invention returns a virtual view of the protected resource for the application program at **308**. In particular, for the protected privilege, an embodiment of the invention generates a copy, if one does not already exist, of the requested resource for read-write access by the application program. In one embodiment in which a copy does not yet exist, a copy is not generated if the request from the application program is only for read access. The copy of the resource is for use only by the application program or group of application programs having the same application identifier. The application identifier allows an embodiment of the invention to provide application programs with different application identifiers their own virtual view or copy of one or more resources. For example, the operating system maintains its own copy of a system setting while an application program writing a value to the system setting receives its own copy of the system setting. In some exemplary embodiments, different applications may receive different virtual views of system settings (e.g., registry entries). Depending on the type of system protection desired (e.g., by a user), a resource may be virtualized per user and/or per application program. Changes to a virtualized resource by an application program with a particular application identifier have no impact (e.g., are not visible) to application programs with other application identifiers. By providing individual applications or groups of applications with their own view of selected system resources, the operating system may prevent one application program from overwriting or otherwise disrupting resources needed by other application programs.

[0051] In one embodiment, an application program uses a virtualized copy of a resource during installation of the application program on a computing system. For example, the application program may apply a system setting to the computing system using a generated copy of a file storing the system setting.

[0052] The application program has access to application private resources. Application private resources include resources that are specific to the application program. The operating system and other application programs are generally unaffected by application private resources. If the application program sends a request to modify an application private resource at **310**, an embodiment of the invention allows and processes the request at **312**.

[0053] The application program may send a request to change system extensibility (e.g., add functionality to the operating system) at **314**. In one embodiment, an embodiment of the invention allows the requested change at **312**.

[0054] Changes to system extensibility and application private resources (e.g., files and system settings) may be logged or otherwise recorded at **318**. Generally, system extensibility changes provide additional functionality to the operating system without modifications to the operating system. Recording the system extensibility changes and changes to application private resources enables the rollback of the changes as well as the complete removal or uninstallation of the application program associated with the changes.

[0055] Example Mitigation Strategy

[0056] Referring next to **FIG. 4**, an exemplary flow chart illustrates operation of a method of providing access control for files, system settings, and extensions. In the example of **FIG. 4**, an operating system implements the method. However, an application program or service not associated with the operating system may also implement the method. In **FIG. 4**, a process is created to execute an application program (e.g., xxxx.exe) via a function such as CreateProcess( ). The operating system determines if there is an application identifier associated with the application program at **402**. If not, the operating system determines the application identifier and persists this information (e.g., stores the determined application identifier in the manifest) at **404**. The application program executes at **406** and performs an operation. The operating system analyzes the operation. For example, in one embodiment, only authorized trusted install processes executing with special privileges may add, modify or delete in protected areas. Application programs are blocked from creating or modifying data in protected areas.

[0057] In the embodiment of **FIG. 4**, if the operation is a file operation at **408**, the operating system determines if the file operation will have an impact on a file (e.g., the file operation modifies the file) at **410**. If the file operation will not have an impact on the file, the operating system allows the file operation to be performed on the file system at **414**. If the file operation will have an impact on the file, the operating system performs a mitigated file operation at **412** according to a mitigation strategy such as illustrated in **FIG. 3**. The change to the file system, if any, is recorded in a log at **415**.

[0058] If the operation is a system setting operation at **416**, the operating system determines if the system setting operation will have an impact on a system setting (e.g., the system setting operation modifies the system setting) at **418**. If the system setting operation will not have an impact on the system setting, the operating system allows the system setting operation to be performed on the system setting at